机器学习攻略

1 局部最小值与鞍点

为什么loss掉不下去？ 走到了loss导数为0的点，此处可能是局部最小值而非全局最小值。local minimal

也可能走到鞍点，在左右上最优，前后上最低这种类型，数据分布形成马鞍的形状。

怎么区别critical point到底是以上的哪一种呢？ 需要知道loss func的形状。

给定一组参数，对Loss泰勒展开，得到:

$$L(\theta) \approx L(\theta') + (\Delta\theta)^T g + 1/2 \cdot (\Delta\theta)^T H(\Delta\theta)$$

而g=0,只需判断后面的一项即可。也即，探讨矩阵H的正定性即可。

如果正定性不确定，则这个点事鞍点。

计算的思路：算出H，然后看正定性，确定这个点在二阶上的性质。

> 探讨：记u为一个特征向量，则式子转化为λu^2，探讨λ

最终发现，只要按照负的特征值方向优化参数，就可以在鞍点的位置继续优化模型。但是这样的方法在实际中 很少使用：算特征向量开销太大力。

那么，还能怎样逃离鞍点?

对比鞍点和局部最优点:

在更高维来看，也许局部最优点是高维空间下的鞍点(这样就有解决的办法了)计算特征值中负值的比例，可以看是否还能继续让loss下降。

2 Batch & Momentum

为啥要提出batch? 所有batch看一遍称为一个epoch

batch的提出能够让模型在一个epoch更新更多次。时间因素随着技术已经没有那么大的区别了。

问题在于这样可能会加入更多噪声的影响。noisy反而更能够训练模型。

为了综合两种方法的优劣，选择一个较为合理的batch_size即可，在引入并行运算之后，实际上一个较大的batch_size未必时间更长。

小的batch甚至会对testing也有帮助!

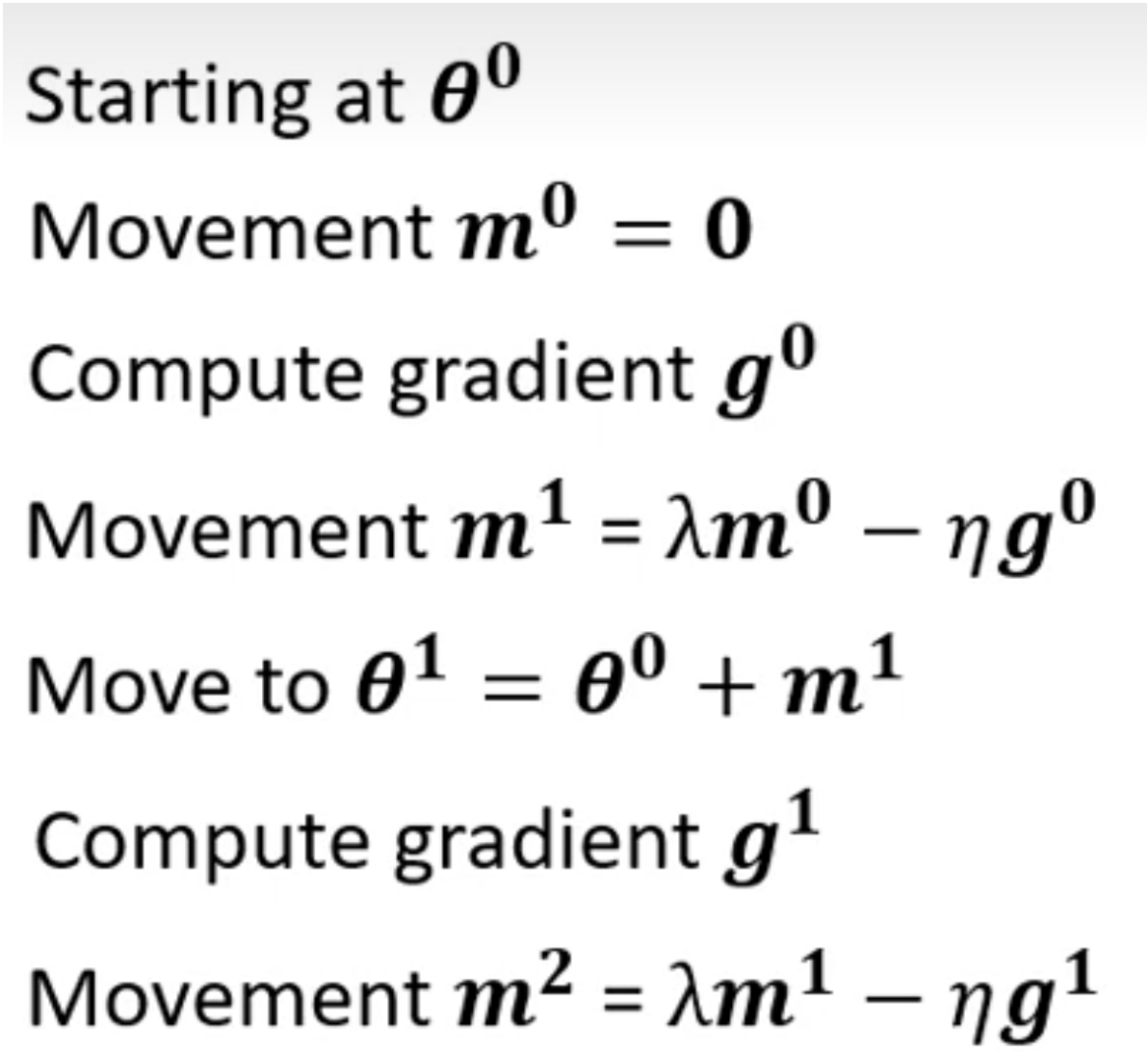
	Small	Large
Speed for one update (no parallel)	Faster	Slower
Speed for one update (with parallel)	Same	Same (not too large)
Time for one epoch	Slower	Faster
Gradient	Noisy	Stable
Optimization	Better	Worse
Generalization	Better	Worse

batch_size 也是一个需要调节的超参数。

思路：参考物理世界的情形，也许可以利用惯性的条件：Momentum技术

Momentum

每次移动参数的时候，和前一步的移动方向加权求和。如图所示：



实际上计算的过程中考虑了之前所有的梯度。

最终可能会形成一个震荡的形式。

3 自动调整学习率

loss不再下降然而gradient依然存在，可能的原因是lr太大力。

真正的情况下，可能走不到 critical point 的时候训练就停止了。

但如果lr太小的话，在gradient太小的地方就很难再减小了。

解决办法：引入一个新的参数使梯度随着训练过程变化。

动态调整: RMSProp

RMSProp

$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} g_i^t$$

$$\theta_i^1 \leftarrow \theta_i^0 - \frac{\eta}{\sigma_i^0} g_i^0 \qquad \sigma_i^0 = \sqrt{\left(g_i^0\right)^2}$$

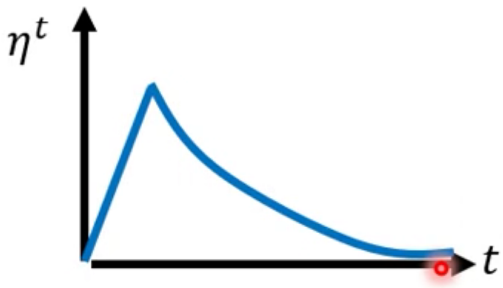
$$0 < \alpha < 1$$

$$\theta_i^2 \leftarrow \theta_i^1 - \frac{\eta}{\sigma_i^1} g_i^1 \qquad \sigma_i^1 = \sqrt{\alpha\left(\sigma_i^0\right)^2 + (1-\alpha)\left(g_i^1\right)^2}$$

调整当前gradient相较于之前所有梯度的比例(重要性)。

另外的方法：1.随着时间增加，让学习率逐渐衰减。

2.warmup:先用大的lr训练一哈，如下：



Warm Up

Increase and then decrease?

Cross Entropy 和Softmax 在pytorch的使用是直接绑定的。

交叉熵相比于最小均方误差在实际应用中更为常用。因为均方误差在优化中很有可能卡在某些位置。